arXiv:0909.3787v2 [cs.FL] 19 Dec 2009

Approximating the minimum length of synchronizing words is hard

Mikhail V. Berlinkov

Department of Algebra and Discrete Mathematics
Ural State University
620083 Ekaterinburg, Russia
berlm@mail.ru

Abstract We prove that, unless P = NP, no polynomial algorithm can approximate the minimum length of synchronizing words for a given synchronizing automaton within a constant factor.

Background and overview

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a complete *deterministic finite automaton* (DFA), where Q is the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is the transition function. The function δ extends in a unique way to a function $Q \times \Sigma^* \to Q$, where Σ^* stands for the free monoid over Σ; the latter function is still denoted by δ. Thus, each word in Σ^* acts on the set Q via δ. The DFA $\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is to leave the automaton in one particular state no matter which state in Q it starts at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$. Any such word w is called a *synchronizing word* for $\mathscr{A}$. The minimum length of synchronizing words for $\mathscr{A}$ is denoted by $\min_{synch}(\mathscr{A})$.

Synchronizing automata serve as transparent and natural models of error-resistant systems in many applications (coding theory, robotics, testing of reactive systems) and also reveal interesting connections with symbolic dynamics and other parts of mathematics. For a brief introduction to the theory of synchronizing automata we refer the reader to the recent survey [11]. Here we discuss only some complexity-theoretical issues of the theory. In the following we assume the reader's acquaintance with some basics of computational complexity that may be found, e.g., in [3,6].

There is a polynomial algorithm (basically due to Černý [1]) that decides whether or not a given DFA is synchronizing. In contrast, determining the minimum length of synchronizing words for a given synchronizing automaton is known to be computationally hard. More precisely, deciding,

given a synchronizing automaton $\mathscr{A}$ and a positive integer ℓ, whether or not $\min_{synch}(\mathscr{A}) \leq \ell$ is NP-complete [2,5,9,8]. Moreover, deciding, given the same instance, whether or not $\min_{synch}(\mathscr{A}) = \ell$ is both NP-hard and co-NP-hard [8]. Thus, unless NP = co-NP, even non-deterministic algorithms cannot find the minimum length of synchronizing words for a given synchronizing automaton in polynomial time.

There are some polynomial algorithms that, given a synchronizing automaton, find synchronizing words for it, see [2,10,7]. Such algorithms can be considered as approximation algorithms for calculating the minimum length of synchronizing words but it seems that they have not been systematically studied from the approximation viewpoint. Experiments show that Eppstein's greedy algorithm [2] behaves rather well on average and approximates $\min_{synch}(\mathscr{A})$ within a logarithmic factor on all tested instances; however, no theoretical justification for these observations has been found so far.

In this paper we prove that, unless P = NP, no polynomial algorithm can approximate the minimum length of synchronizing words for a given synchronizing automaton within a constant factor. This result was announced in the survey [11] (with a reference to the present author's unpublished manuscript) but its proof appears here for the first time. We also mention that a special case of our result, namely, non-approximability of $\min_{synch}(\mathscr{A})$ within factor 2, was announced by Gawrychowski [4].

The paper is organized as follows. First we exhibit an auxiliary construction that shows non-approximability of $\min_{synch}(\mathscr{A})$ within factor $2-\varepsilon$ for automata with 3 input letters. Then we show how to iterate this construction in order to obtain the main result, again for automata with 3 input letters. Finally, we describe how the construction can be modified to extend the result also to automata with only 2 input letters.

1 Non-approximability within factor $2 - \varepsilon$

First we fix our notation and introduce some definitions. When we have specified a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, we can simplify the notation by writing $q.w$ instead of $\delta(q, w)$ for $q \in Q$ and $w \in \Sigma^*$. For each subset $S \subseteq Q$ and each word $w \in \Sigma^*$, we write $S.w$ instead of $\{q.w \mid q \in S\}$. We say that a subset $S \subseteq Q$ is *occupied* after applying some word $v \in \Sigma^*$ if $S \subseteq Q.v$.

The length of a word $w \in \Sigma^*$ is denoted by $|w|$. If $1 \leq s \leq |w|$, then $w[s]$ denotes the letter in the s-th position of w; similarly, if $1 \leq s < t \leq |w|$, then $w[s..t]$ stands for the word $w[s]w[s+1]\cdots w[t]$.

Let $\mathcal{K}$ be a class of synchronizing automata We say that an algorithm M *approximates the minimal length of synchronizing words in* $\mathcal{K}$ if, for an arbitrary DFA $\mathscr{A} \in \mathcal{K}$, the algorithm calculates a positive integer $M(\mathscr{A})$ such that $M(\mathscr{A}) \geq \min_{synch}(\mathscr{A})$. The *performance ratio* of M at $\mathscr{A}$ is $R_M(\mathscr{A}) = \dfrac{M(\mathscr{A})}{\min_{synch}(\mathscr{A})}$. The algorithm is said to *approximate the minimal length of synchronizing words within factor* $k \in \mathbb{R}$ if

$$\sup\{R_M(\mathscr{A}) \mid \mathscr{A} \in \mathcal{K}\} = k.$$

Even though the following theorem is subsumed by our main result, we prove it here because the proof demonstrates underlying ideas in a nutshell and in the same time presents a construction that serves as the induction basis for the proof of the main theorem.

Theorem 1. *If* P $\neq$ NP, *then for no* $\varepsilon > 0$ *a polynomial algorithm approximates the minimal length of synchronizing words within factor* $2-\varepsilon$ *in the class of all synchronizing automata with* 3 *input letters.*

Proof. Arguing by contradiction, assume that there exist a real number $\varepsilon > 0$ and a polynomial algorithm M such that $R_M(\mathscr{A}) \leq 2-\varepsilon$ for every synchronizing automaton $\mathscr{A}$ with 3 input letters.

We fix an arbitrary $n > 2$ and take an arbitrary instance ψ of the classical NP-complete problem SAT (the satisfiability problem for a system of clauses, that is, formulae in conjunctive normal form) with n variables. Let m be the number of clauses in ψ. We shall construct a synchronizing automaton $\mathscr{A}(\psi)$ with 3 input letters and polynomial in m, n number of states such that $\min_{synch}(\mathscr{A}(\psi)) = n + 2$ if ψ is satisfiable and $\min_{synch}(\mathscr{A}(\psi)) > 2(n-1)$ if ψ is not satisfiable. If n is large enough, namely, $n \geq \frac{6}{\varepsilon} - 2$, then we can decide whether or not ψ is satisfiable by running the algorithm M on $\mathscr{A}(\psi)$. Indeed, if ψ is not satisfiable, then $M(\mathscr{A}(\psi)) \geq \min_{synch}(\mathscr{A}(\psi)) > 2(n-1)$, but, if ψ is satisfiable, then

$$\begin{aligned} M(\mathscr{A}(\psi)) \leq (2-\varepsilon)\min_{synch}(\mathscr{A}(\psi)) &= (2-\varepsilon)(n+2) \\ &\leq (2 - \frac{6}{n+2})(n+2) = 2(n-1). \end{aligned}$$

Clearly, this yields a polynomial algorithm for SAT: given an instance of SAT, we can first, if necessary, enlarge the number of variables to at least $\frac{6}{\varepsilon} - 2$ without influencing satisfiability and then apply the above procedure. This contradicts the assumption that P $\neq$ NP.

Now we describe the construction of the automaton $\mathscr{A}(\psi) = \langle Q, \Sigma, \delta \rangle$. The state set Q of $\mathscr{A}(\psi)$ is the disjoint union of the three following sets:

$$S^1 = \{q_{i,j} \mid 1 \leq i \leq m+1,\ 1 \leq j \leq n+1,\ i \neq m+1 \text{ or } j \neq n+1\},$$
$$S^2 = \{p_{i,j} \mid 1 \leq i \leq m+1,\ 1 \leq j \leq n+1\},$$
$$S^3 = \{z_1, z_0\}.$$

The size of Q is equal to $2(m+1)(n+1)+1$ so a polynomial in m, n.

The input alphabet Σ of $\mathscr{A}(\psi)$ is the set $\{a, b, c\}$. In order to describe the transition function $\delta : Q \times \Sigma \to Q$, we need an auxiliary function $f : \{a, b\} \times \{1, \ldots, m\} \times \{1, \ldots, n\} \to Q$ defined as follows. Let the variables involved in ψ be $x_1, \ldots, x_n$ and the clauses of ψ be $c_1, \ldots, c_m$. For a literal $y \in \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$ and a clause c_i, we write $y \in c_i$ to denote that y appears in c_i. Now set

$$f(d,i,j) = \begin{cases} z_0 \text{ if } d = a \text{ and } x_j \in c_i, \\ z_0 \text{ if } d = b \text{ and } \neg x_j \in c_i, \\ q_{i,j+1} \text{ otherwise.} \end{cases}$$

The transition function δ is defined according to the following table:

State $q \in Q$	$\delta(q,a)$	$\delta(q,b)$	$\delta(q,c)$
$q_{i,j}$ for $1 \leq i \leq m, 1 \leq j \leq n$	$f(a,i,j)$	$f(b,i,j)$	$q_{i,1}$
$q_{m+1,j}$ for $1 \leq j \leq n$	$q_{m+1,j+1}$	$q_{m+1,j+1}$	$q_{m+1,1}$
$q_{i,n+1}$ for $1 \leq i \leq m$	z_0	z_0	$q_{m+1,1}$
$p_{i,j}$ for $1 \leq i \leq m+1, 1 \leq j \leq n$	$p_{i,j+1}$	$p_{i,j+1}$	$p_{i,j+1}$
$p_{i,n+1}$ for $1 \leq i \leq m+1$	z_0	z_0	$q_{i,1}$
z_1	$q_{m+1,1}$	$q_{m+1,1}$	z_0
z_0	z_0	z_0	z_0

Let us informally comment on the essence of the above definition. Its most important feature is that, if the literal x_j (respectively $\neg x_j$) occurs in the clause c_i, then the letter a (respectively b) moves the state $q_{i,j}$ to the state z_0. This encodes the situation when one can satisfy the clause c_i by choosing the value 1 (respectively 0) for the variable x_j. Otherwise, the letter a (respectively b) increases the second index of the state. This means that one cannot make c_i be true by letting $x_j = 1$ (respectively $x_j = 0$), and the next variable has to be inspected. Of course, this encoding idea is not new, see, e.g., [2].

By the definition, z_0 is the zero state of the automaton $\mathscr{A}(\psi)$. Since there is a path to z_0 from each state $q \in Q$, the automaton $\mathscr{A}(\psi)$ is synchronizing.

Figure 1 shows two automata of the form $\mathscr{A}(\psi)$ build for the SAT instances

$$\psi_1 = \{x_1 \vee x_2 \vee x_3,\ \neg x_1 \vee x_2,\ \neg x_2 \vee x_3,\ \neg x_2 \vee \neg x_3\},$$
$$\psi_2 = \{x_1 \vee x_2,\ \neg x_1 \vee x_2,\ \neg x_2 \vee x_3,\ \neg x_2 \vee \neg x_3\}.$$

If at some state $q \in Q$ the picture has no outgoing arrow labelled $d \in \Sigma$, the arrow $q \xrightarrow{d} z_0$ is assumed (all those arrows are omitted in the picture to improve readability). The two instances differ only in the first clause: in ψ_1 it contains the variable x_3 while in ψ_2 it does not. Correspondingly, the automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$ differ only by the outgoing arrow labelled a at the state $q_{1,3}$: in $\mathscr{A}(\psi_1)$ it leads to z_0 (and therefore, it is not shown) while in $\mathscr{A}(\psi_2)$ it leads to the state $q_{1,4}$ and is shown by the dashed line.

Observe that ψ_1 is satisfiable for the truth assignment $x_1 = x_2 = 0$, $x_3 = 1$ while ψ_2 is not satisfiable. It is not hard to check that the word $cbbac$ synchronizes $\mathscr{A}(\psi_1)$ and the word a^7c is one of the shortest reset words for $\mathscr{A}(\psi_2)$.

To complete the proof, it remains to show that $\min_{synch}(\mathscr{A}(\psi)) = n+2$ if ψ is satisfiable and $\min_{synch}(\mathscr{A}(\psi)) > 2(n-1)$ if ψ is not satisfiable. First consider the case when ψ is satisfiable. Then there exists a truth assignment $\tau : \{x_1, \ldots, x_n\} \to \{0,1\}$ such that $c_i(\tau(x_1), \ldots, \tau(x_n)) = 1$ for every clause c_i of ψ. We construct a word $v = v(\tau)$ of length n as follows:

$$v[j] = \begin{cases} a & \text{if } \tau(x_j) = 1, \\ b & \text{if } \tau(x_j) = 0. \end{cases} \tag{1}$$

We aim to prove that the word $w = cvc$ is a synchronizing word for $\mathscr{A}(\psi)$, that is, $Q.w = \{z_0\}$. Clearly, $z_1.c = z_0$. Further, $S^2.cv = \{z_0\}$ because every word of length $n+1$ that does not end with c sends S^2 to z_0. Now let $T = \{q_{i,1} \mid 1 \le i \le m+1\}$, so T is the "first row" of S^1. Observe that $S^1.c = T$. Since $c_i(\tau(x_1), \ldots, \tau(x_n)) = 1$ for every clause c_i, there exists an index j such that either $x_j \in c_i$ and $\tau(x_j) = 1$ or $\neg x_j \in c_i$ and $\tau(x_j) = 0$. This readily implies (see the comment following the definition of the transition function of $\mathscr{A}(\psi)$) that $q_{i,1}.v = z_0$ for all $1 \le i \le m$. On the other hand, $q_{m+1,1}.v = z_1$ because every word of length n that does not involve c sends $q_{m+1,1}$ to z_1. Thus, $S^1.cv = T.v = S^3$ and $S^1.w = \{z_0\}$. We have shown that w synchronizes $\mathscr{A}(\psi)$, and it is clear that $|w| = n+2$ as required.

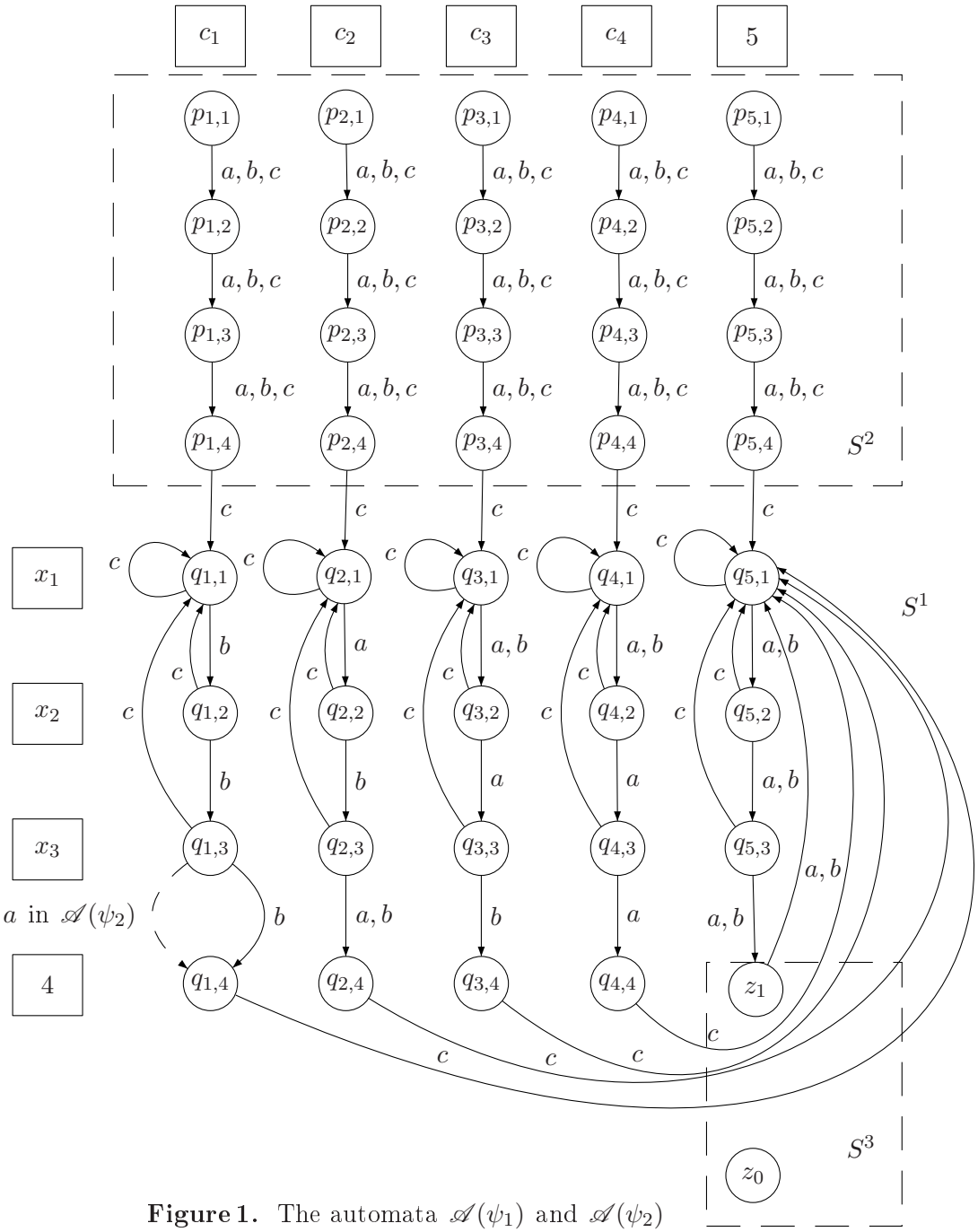

Figure 1. The automata $\mathscr{A}(\psi_1)$ and $\mathscr{A}(\psi_2)$

Now we consider the case when ψ is not satisfiable.

Lemma 1. *If ψ is not satisfiable, then, for each word $v \in \{a,b\}^*$ of length n, there exists $i \leq m$ such that $q_{i,n+1} \in T.v$.*

Proof. Define a truth assignment $\tau : \{x_1, \ldots, x_n\} \to \{0,1\}$ as follows:

$$\tau(x_j) = \begin{cases} 1 & \text{if } v[j] = a, \\ 0 & \text{if } v[j] = b. \end{cases}$$

Since ψ is not satisfiable, we have $c_i(\tau(x_1), \ldots, \tau(x_n)) = 0$ for some clause c_i, $1 \leq i \leq m$. According to our definition of the transition function of $\mathscr{A}(\psi)$, this means that $q_{i,j}.v[j] = q_{i,j+1}$ for all $j = 1, \ldots, n$. Hence $q_{i,n+1} = q_{i,1}.v \in T.v$. □

Lemma 2. *If ψ is not satisfiable, then for each word $v \in \{a,b\}^*$ of length n and each letter $d \in \Sigma$, the state $q_{m+1,1}$ belongs to $T.vd$.*

Proof. If $d = c$, the claim follows from Lemma 1 and the equalities $q_{m+1,1} = q_{i,n+1}.c$ that hold for all $i \leq m$. If $d \neq c$, we observe that the state $q_{m+1,1}$ is fixed by all words of length $n+1$ not involving c. □

Let w' be a synchronizing word of minimal length for $\mathscr{A}(\psi)$ and denote $w = cw'c$. Then the word w is also synchronizing and $\ell = |w| > n$ because already the length of the shortest path from $q_{m+1,1}$ to z_0 is equal to $n+1$. Let k be the rightmost position of the letter c in the word $w[1..n]$.

Lemma 3. $T \subseteq Q.w[1..k]$.

Proof. Indeed, since $k \leq n$, for each $1 \leq i \leq m+1$ we have

$$p_{i,n+2-k}.w[1..k-1]w[k] = p_{i,n+1}.c = q_{i,1} \in T.$$ □

We denote by v the longest prefix of the word $w[k+1..\ell]$ such that $v \in \{a,b\}^*$ and $|v| \leq n$. Since w ends with c, the word v cannot be a suffix of w. Let $d \in \Sigma$ be the letter that follows v in w. If $|v| = n$, then Lemma 2 implies that $q_{m+1,1} \in T.vd$. If $|v| < n$, then by the definition of v we have $d = c$. Hence

$$q_{m+1,1}.vd = q_{m+1,|v|+1}.c = q_{m+1,1}.$$

Thus, $q_{m+1,1} \in T.vd$ also in this case. Combining this with Lemma 3, we have

$$Q.w[1..k]vd \supseteq T.vd \ni q_{m+1,1}. \tag{2}$$

From the definitions of k and v it readily follows that $w[k+1..n]$ is a prefix of v whence $|v| \geq n-k$. Thus, $|w[1..k]vd| \geq k+(n-k)+1 = n+1$. Recall that the length of the shortest path from $q_{m+1,1}$ to z_0 is equal to $n+1$, and the suffix of w following $w[1..k]vd$ must bring the state $q_{m+1,1}$ to z_0 in view of (2). Hence $|w| \geq (n+1)+(n+1) = 2n+2 > 2n$ and $|w'| > 2(n-1)$. We have proved that $\min_{synch}(\mathscr{A}(\psi)) > 2(n-1)$ if ψ is not satisfiable. □

2 The main result

The main result of this paper is

Theorem 2. *If* P ≠ NP*, then no polynomial algorithm can approximate the minimal length of synchronizing words within a constant factor in the class of all synchronizing automata with* 3 *input letters.*

Proof. Again we fix an arbitrary $n > 2$ and take an arbitrary instance ψ of SAT with n variables. We shall prove by induction that for every $r = 2, 3, \ldots$ there exists a synchronizing automaton $\mathscr{A}_r(\psi) = \langle Q_r, \Sigma, \delta_r \rangle$ with the following properties:

- $\Sigma = \{a, b, c\}$;
- $|Q_r|$ is bounded by a polynomial of n and the number m of clauses of ψ;
- if ψ is satisfiable under a truth assignment $\tau : \{x_1, \ldots, x_n\} \to \{0, 1\}$, then the word $w = c^{r-1}v(\tau)c$ of length $n+r$ synchronizes $\mathscr{A}_r(\psi)$ (see (1) for the definition of the word $v(\tau)$);
- $\min_{synch}(\mathscr{A}_r) > r(n-1)$ if ψ is not satisfiable.

Then, applying the same standard argument as in the proof of Theorem 1, we conclude that for no $\varepsilon > 0$ the minimal length of synchronizing words can be approximated by a polynomial algorithm within factor $r-\varepsilon$. Since r can be arbitrarily large, the statement of the main result follows.

The induction basis is verified in the proof of Theorem 1: we can choose the synchronizing automaton $\mathscr{A}(\psi)$ to play the role of $\mathscr{A}_2(\psi)$. For the sake of uniformity, in the sequel we refer to the state set Q of $\mathscr{A}(\psi)$ and its transition function δ as to Q_2 and respectively δ_2.

Now suppose that $r > 2$ and the automaton $\mathscr{A}_{r-1}(\psi) = \langle Q_{r-1}, \Sigma, \delta_{r-1} \rangle$ with the desired properties has already been constructed. We let

$$Q_r = Q_{r-1} \bigcup (Q_2 \setminus \{z_0\}) \times Q_{r-1}.$$

Clearly, $|Q_r| = |Q_{r-1}| \cdot |Q_2|$ and from the induction assumption it follows that $|Q_r|$ is a polynomial in m, n.

We now define the transition function $\delta_r : Q_r \times \Sigma \to Q_r$. Let $d \in \Sigma$, $q \in Q_r$. If $q \in Q_{r-1}$, then we set

$$\delta_r(q, d) = \delta_{r-1}(q, d). \tag{3}$$

If $q = (q', q'') \in (Q_2 \setminus \{z_0\}) \times Q_{r-1}$, we define

$$\delta_r(q,d) = \begin{cases} z_0 & \text{if } \delta_2(q',d) = z_0, \\ q'' & \text{if } \delta_2(q',d) = q_{m+1,1} \text{ and either} \\ & q' = q_{i,n+1} \text{ for } i \in \{1,\dots,m\} \\ & \text{or } q' = q_{m+1,j} \text{ for } j \in \{2,\dots,n\} \\ & \text{or } q' = z_1, \\ (\delta_2(q',d), q'') & \text{in all other cases.} \end{cases} \tag{4}$$

Using this definition and the induction assumption, one can easily verify that the state z_0 is the zero state of the automaton $\mathscr{A}_r(\psi)$ and that there is a path to z_0 from every state in Q_r. Thus, $\mathscr{A}_r(\psi)$ is a synchronizing automaton.

In order to improve readability, we denote the subset $\{q_{i,j}\} \times Q_{r-1}$ by $Q_{i,j}$ for each state $q_{i,j} \in S^1$ and the subset $\{p_{i,j}\} \times Q_{r-1}$ by $P_{i,j}$ for each state $p_{i,j} \in S^2$. Slightly abusing notation, we denote by T the "first row" of $S^1 \times Q_{r-1}$, i.e. $T = \bigcup_{1 \le i \le m+1} Q_{i,1}$. Similarly, let $P = \bigcup_{1 \le i \le m+1} P_{i,1}$ be the "first row" of $S^2 \times Q_{r-1}$. We also specify that the dot-notation (like $q.d$) always refers to the function δ_r.

First we aim to show that if ψ is satisfiable under a truth assignment $\tau : \{x_1, \dots, x_n\} \to \{0,1\}$, then the word $w = c^{r-1}v(\tau)c$ synchronizes the automaton $\mathscr{A}_r(\psi)$. By (3) and the induction assumption we have $Q_{r-1}.c \subseteq Q_{r-1}$ and $Q_{r-1}.c^{r-2}v(\tau)c = z_0$. Further, we can decompose $((Q_2 \setminus \{z_0\}) \times Q_{r-1}).c$ as $\{z_0\} \cup F_{r-1} \cup F_r$ for some sets $F_{r-1} \subseteq Q_{r-1}$ and $F_r \subseteq (Q_2 \setminus \{z_0\}) \times Q_{r-1}$. By the induction assumption,

$$F_{r-1}.c^{r-2}v(\tau)c \subseteq Q_{r-1}.c^{r-2}v(\tau)c = z_0$$

Consider the set F_r. Using the definition of the action of c on Q_2 via δ_2, one can observe that $F_r = T \cup G$ where G stands for $S^2 \times Q_{r-1} \setminus P$. From (4) we see that $T.c = T$ and $G.c \subseteq T \cup G$. Thus we have $F_r.c^{r-2}v(\tau)c \subseteq T.v(\tau)c \cup G.v(\tau)c$, and combining the first alternative in (4) with properties of the automaton $\mathscr{A}_2(\psi)$ established in the proof of Theorem 1, we obtain $T.v(\tau)c = G.v(\tau)c = \{z_0\}$.

Now we consider the case when ψ is not satisfiable. The following lemma is parallel to Lemma 1 and has the same proof because the action of a and b on the "blocks" $Q_{i,j}$ with $1 \leq i \leq m$ and $1 \leq j \leq n$ via δ_r precisely imitates the action of a and b on the states $q_{i,j}$ in the automaton $\mathscr{A}(\psi)$, see the last alternative in (4).

Lemma 4. *If ψ is not satisfiable, then, for each word $v \in \{a,b\}^*$ of length n, there exists $i \leq m$ such that $Q_{i,n+1} \subseteq \delta_r(T,v)$.* □

In contrast, the next lemma which is a counterpart of Lemma 2 uses the fact that in some cases the action of the letters via δ_r drops states from $((Q_2 \setminus z_0) \times Q_{r-1})$ down to Q_{r-1}, see the middle alternative in (4).

Lemma 5. *If ψ is not satisfiable, then for each word $v \in \{a,b\}^*$ of length n and each letter $d \in \Sigma$, we have $Q_{r-1} \subseteq \delta_r(T,vd)$.*

Proof. If $d = c$, the claim follows from Lemma 4 and the equalities $\delta_r((q_{i,n+1},q''),c) = q''$ that hold for all $i \leq m$ and all $q'' \in Q_{r-1}$. If $d \neq c$, we observe that $\delta_r((q_{m+1,1},q''),v) = (z_1,q'')$ and $\delta_r((z_1,q''),a) = \delta_r((z_1,q''),b) = q''$ for all $q'' \in Q_{r-1}$. □

Let w' be a synchronizing word of minimal length for $\mathscr{A}_r(\psi)$ and denote $w = cw'c$. Then the word w is also synchronizing and $\ell = |w| > (r-1)n$ by the induction assumption. Let k be the rightmost position of the letter c in the word $w[1..n]$. We have the next lemma parallel to Lemma 3 and having the same proof (with the "blocks" $P_{i,j}$ with $1 \leq i \leq m+1$, $n+2-k \leq j \leq n$ playing the role of the states $p_{i,j}$).

Lemma 6. $T \subseteq \delta_r(Q_r, w[1..k])$. □

Now, as in the proof of Theorem 1, we denote by v the longest prefix of the word $w[k+1..\ell]$ such that $v \in \{a,b\}^*$ and $|v| \leq n$. Clearly, v cannot be a suffix of w. Let $d \in \Sigma$ be the letter that follows v in w. If $|v| = n$ then Lemma 5 implies that $Q_{r-1} \subseteq \delta_r(T,vd)$. If $|v| < n$, then by the definition of v we have $d = c$. Hence

$$\delta_r(Q_{m+1,1},vd) = \delta_r(Q_{m+1,|v|+1},c) = Q_{r-1}.$$

Thus, $Q_{r-1} \subseteq \delta_r(T,vd)$ also in this case. Combining this with Lemma 6, we have

$$\delta_r(Q_r, w[1..k]vd) \supseteq \delta_r(T,vd) \supseteq Q_{r-1}. \tag{5}$$

From the definitions of k and v it readily follows that $|v| \geq n-k$. Thus, $|w[1..k]vd| \geq k + (n-k) + 1 = n+1$. The suffix of w following

$w[1..k]vd$ must bring the set Q_{r-1} to a single state in view of (5). However, by (3) the restriction of δ_r to Q_{r-1} coincides with δ_{r-1} whence the suffix must be a synchronizing word for $\mathscr{A}_{r-1}(\psi)$. By the induction assumption $\min_{synch}(\mathscr{A}_{r-1}(\psi)) > (r-1)(n-1)$, and therefore,

$$|w| > (n+1) + (r-1)(n-1) = r(n-1) + 2$$

and $|w'| > r(n-1)$. We have thus proved that $\min_{synch}(\mathscr{A}_r(\psi)) > r(n-1)$ if ψ is not satisfiable. This completes the induction step. □

3 The case of 2-letter alphabets

We show that the main result extends to synchronizing automata with only 2 input letters.

Corollary 1. *If* P ≠ NP*, then no polynomial algorithm can approximate the minimal length of synchronizing words within a constant factor in the class of all synchronizing automata with* 2 *input letters.*

Proof. For any synchronizing automaton $\mathscr{A} = (Q, \{a_1, a_2, a_3\}, \delta)$ we can construct a synchronizing automaton $\mathscr{B} = (Q', \{a, b\}, \delta')$ such that

$$\min_{synch}(\mathscr{A}) \leq \min_{synch}(\mathscr{B}) \leq 3 \min_{synch}(\mathscr{A}) \tag{6}$$

and $|Q'|$ is a polynomial of $|Q|$. Then any polynomial algorithm approximating the minimal length of synchronizing words for 2-letter synchronizing automata within factor r would give rise to a polynomial algorithm approximating the minimal length of synchronizing words for 3-letter synchronizing automata within factor $3r$. This would contradict Theorem 2.

We let $Q' = Q \times \{a_1, a_2, a_3\}$ and define the transition function $\delta' : Q' \times \{a, b\} \to Q'$ as follows:

$$\delta'((q, a_i), a) = (q, a_{\min(i+1,3)}),$$
$$\delta'((q, a_i), b) = (\delta(q, a_i), a_1).$$

Thus, the action of a on a state $q' \in Q'$ substitutes an appropriate letter from in the alphabet $\{a_1, a_2, a_3\}$ of $\mathscr{A}$ for the second component of q' while the action of b imitates the action of the second component of q' on its first component and resets the second component to a_1. Now is let a word $w \in \{a_1, a_2, a_3\}$ of length ℓ be a synchronizing word for $\mathscr{A}$. Define

$$v_s = \begin{cases} b & \text{if } w[s] = a_1, \\ ab & \text{if } w[s] = a_2, \\ aab & \text{if } w[s] = a_3. \end{cases}$$

Then the word $v = bv_1 \cdots v_\ell$ is easily seen to be a synchronizing word for $\mathscr{B}$ and $|v| \leq 3\ell$ unless all letters in w are a_3 but in this case we can just let a_2 and a_3 swap their names. Hence the second inequality in (6) holds true, and the first inequality is clear.

Acknowledgments

The author acknowledges support from the Federal Education Agency of Russia, grant 2.1.1/3537, and from the Russian Foundation for Basic Research, grant 09-01-12142.

References

1. Černý, J.: Poznámka k homogénnym eksperimentom s konečnými automatami. Matematicko-fyzikálny Časopis Slovensk. Akad. Vied 14(3) 208–216 (1964) (in Slovak)
2. Eppstein, D.: Reset sequences for monotonic automata. SIAM J. Comput. 19, 500–510 (1990)
3. Garey, M. R.; Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco (1979)
4. Gawrychowski, P.: Complexity of the approximation of the shortest synchronizing word. In: Workshop "Around the Černý Conjecture". Univ. Wrocław, 2008 (unpublished)
5. Goralčik, P.; Koubek, V. Rank problems for composite transformations. Int. J. Algebra and Computation 5, 309–316 (1995)
6. Papadimitriou, C. H.: Computational Complexity, Addison-Wesley, Reading, MA (1994)
7. Roman, A.: Synchronizing finite automata with short reset words. Appl. Math. and Computation 209, 125Ц-136 (2009)
8. Samotij, W.: A note on the complexity of the problem of finding shortest synchronizing words. In: Electronic Proc. AutoMathA 2007, Automata: from Mathematics to Applications. Univ. Palermo, Palermo (2007)
9. Salomaa, A.: Composition sequences for functions over a finite domain. Theoret. Comp. Sci. 292, 263–281 (2003)
10. Trahtman, A.: An efficient algorithm finds noticeable trends and examples concerning the Černý conjecture. In: Královič, R.; Urzyczyn, P. (eds.), 31st Int. Symp. Math. Foundations of Comput. Sci. Lect. Notes Comput. Sci., vol. 4162, pp. 789–800. Springer, Heidelberg (2006)
11. Volkov, M.V.: Synchronizing automata and the Černý conjecture. In: Martín-Vide, C.; Otto, F.; Fernau, H. (eds.) Languages and Automata: Theory and Applications. Lect. Notes Comput. Sci., vol. 5196, pp. 11–27. Springer, Heidelberg (2008)